Application

for

United States Patent

To all whom it may concern:

Be it known that, Joseph Horanzy and Akshay Mathur

have invented certain new and useful improvements in

A METHOD AND APPARATUS FOR BOOTING A MICROPROCESSOR

of which the following is a full, clear and exact description:

A METHOD AND APPARATUS FOR BOOTING A MICROPROCESSOR

FIELD OF THE INVENTION

[0001] The present invention relates generally to boot-up procedures for a processor. More particularly, the present invention relates to booting-up a processor from a volatile memory device.

BACKGROUND OF THE INVENTION

[0002] Computers and computer-based devices such as personal computers (PCs), personal digital assistants (PDAs) and other embedded devices currently employ large, often complex, operating systems to execute user programs and process data. Conventional operating systems typically range in size from hundreds of kilobytes for small PDAs to hundreds of megabytes for high-end servers and PCs. To initialize or reset a computer or processor, a boot process is performed using system software and/or firmware.

[0003] Firmware refers to processor routines that are stored in non-volatile structures such as read only memories (ROMs), flash memories, and the like. These memory structures preserve the code stored in them, even when power is turned off. One of the principle uses of firmware is to provide routines that control a computer system when it is powered up from a shut down state, before volatile memory structures have been tested and configured. The process by which a computer is brought to its operating state from a powered down or powered off state is referred to as bootstrapping. Firmware routines may also be used to reinitialize or

reconfigure the computer system following various hardware events and to handle certain platform level events like system interrupts.

[0004] The bootstrapping process typically begins with a processor(s) in a computer system and proceeds outward to system level resources. Initially, each processor tests its internal components and interfaces. In multi-processor systems, a single bootstrap processor is usually selected to handle initialization procedures for the system as a whole. These procedures include checking the integrity of memory, identifying and initializing other resources in the computer system, loading the operating system into memory, and initializing the remaining processors. Since volatile memory structures such as caches and random access memory (RAM) are not dependable until later in the boot process, the processor implements some of its early firmware routines for the various bootstrapping procedures inside nonvolatile memory such as flash devices. One advantage of using flash devices is that they can be quickly reprogrammed without being removed form the system. The flash memory, by virtue of its electrical programmability and erasability, is easily updated under system software control while being physically connected to the computer motherboard.

[0005] Typically, many processors in a computer system may have their own flash devices for storing, amongst other things, bootstrapping procedures. One problem with these systems is the cost and board space needed to provide all of these flash devices including their support hardware and sockets. Another problem may occur if the bootstrapping procedure needs to be changed. If the bootstrapping

procedure is changed, a user must make sure that the procedure has been changed in all of the flash devices, or else the system may not initializes correctly.

[0006] Accordingly, it is desirable to provide a method and apparatus for bootstrapping a plurality of processors from a single flash device so as to save costs and board space while ensuring the ease of updating changes to the bootstrapping procedure.

SUMMARY OF THE INVENTION

[0007] In one aspect of the present invention a method and apparatus is provided for allowing a processor to boot from a volatile memory device, thus eliminating the need for some flash devices in a computer system.

[0008] In another aspect of the present invention a method and apparatus is provided for allowing a user to select between booting a processor from a flash device or a volatile memory device when both options are available.

[0009] In yet another aspect of the invention a novel method and apparatus for tricking the processor to boot from a volatile memory device is disclosed. In accordance with one embodiment of the present invention, a method and system for bootstrapping a processor from a volatile memory device associated with the processor is disclosed. The master processor is bootstrapped from flash device. The reset lines of the processor are asserted. The boot code for the processor is loaded from the flash device into the volatile memory device. The reset lines of the

processor are de-asserted, wherein the processor will then boot from the boot code stored in the volatile memory device.

[0010] In accordance with another aspect of the present invention, a method and system for bootstrapping a processor from either a flash device or a volatile memory device is disclosed. The master processor is first bootstrapped. The master processor then determines whether a flash bootstrap procedure or a volatile memory device bootstrap procedure has been selected for the microprocessor. A command register in a logic device is set to configure logic units for the selected bootstrap procedure. If the flash bootstrap procedure was selected, the bootstrap procedure from the flash device associated with the processor is performed. However, if the volatile memory bootstrap procedure was selected, the reset lines of the second processor are asserted. The boot code for the processor in the flash device connected to the master processor is loaded into the volatile memory device of the second processor. The reset lines of the second processor are de-asserted, wherein the processor will then boot from the boot code stored in the volatile memory device.

[0011] There has thus been outlined, rather broadly, the more important features of the invention in order that the detailed description thereof that follows may be better understood, and in order that the present contribution to the art may be better appreciated. There are, of course, additional features of the invention that will be described below and which will form the subject matter of the claims appended hereto.

[0012] In this respect, before explaining at least one embodiment of the invention in detail, it is to be understood that the invention is not limited in its application to the details of construction and to the arrangements of the components set forth in the following description or illustrated in the drawings. The invention is capable of other embodiments and of being practiced and carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein, as well as the abstract, are for the purpose of description and should not be regarded as limiting.

[0013] As such, those skilled in the art will appreciate that the conception upon which this disclosure is based may readily be utilized as a basis for the designing of other structures, methods and systems for carrying out the several purposes of the present invention. It is important, therefore, that the claims be regarded as including such equivalent constructions insofar as they do not depart from the spirit and scope of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] FIG. 1 provides a block diagram of a computer system of a preferred embodiment of the present invention.

[0015] FIG. 2 is a flowchart illustrating the steps that may be followed in accordance with one embodiment of the present inventive method or process.

[0016] FIG. 3 provides a block diagram of a computer system of a preferred embodiment of the present invention.

[0017] FIG. 4 is a flowchart illustrating the steps that may be followed in accordance with one embodiment of the present inventive method or process.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

[0018] A preferred embodiment of the present invention provides a system and method for performing bootstrap procedures from volatile memory devices, such as static random access memory (SRAM) or dual port random access memory (DPRAM). The present invention "tricks" processors into believing that they are being bootstrapped by flash devices, when in fact they are being bootstrapped by volatile memory devices.

[0019] A preferred embodiment of the present inventive apparatus and method is illustrated in FIG. 1, which illustrates a computer system 100. The computer system 100 is comprised of a master processing unit 102, a flash device 106, a programmable logic device 108, a local data bus 110, a plurality of slave processing units 112, 122, volatile memory devices 114, 124 and a plurality of data busses 116, 126. As illustrated by the dashed lines, any number of slave processing units and associated devices can be connected to the computer system 100 and the present invention is not limited to the two slave processing units illustrated in FIG. 1.

[0020] The first processing unit 102 is connected to the flash device(s) 106 and a communications port 104. While the description of the embodiments of the

present invention makes reference to processors and processing units, it will be understood that the description pertains to processors, microprocessors, processing units and the like and the invention is not limited thereto. The first processing unit 102 is also connected to the local data bus 110, which is connected to the logic device 108 and the plurality of volatile memory devices 114, 124. The programmable logic device 108 is a programmable integrated circuit that allows the user of the circuit, using software control, to customize the logic functions the circuit will perform. In this case, the programmable logic device is controlled by the first processing unit 102 to provide output enable signals to the volatile memory devices and reset control lines to the processors 112 and 122. The programmable logic device also receives volatile/non-volatile output enable control signals from each second processing unit 112, 122. The programmable logic device 108 can be a complex programmable logic device (CPLD), a field programmable gate array (FPGA), or the like. Each second processing unit 112, 122 is connected to its associated volatile memory device 114, 124 by the data busses 116 and 126, respectively. The volatile memory devices can be static random access memory devices, dual port random access memory devices, or the like. A pulldown network 119, 129 is connected between each data bus 116, 126 and ground. This resistor network insures that the data bus is at a zero state when the bus is in tristate mode. Thus, when the second processor reads the data bus at a zero state, it is a no-op command to the processor. This method is needed for synchronous volatile memory since the processor during power-up does not supply the clock for about 10

execution cycles that is needed to read data from synchronous memory. The no-op command is a 'filler' for this interval until the processor produces the clock.

[0021] The operation of the computer system 100 will now be described with reference to FIG. 2. In order to boot the slave processing units 112 and 122 from the volatile memory devices 114 and 124, respectively, the programmable logic device 108 contains logic to correctly combine the second processing unit's memory interface signals and the hardware handshake signals of the volatile memory device 114, 124 to 'trick' the slave processing units 112, 122 into thinking that they are being bootstrapped using flash devices. First, the first processing unit 102 is bootstrapped in a known manner using the flash device 106. During system initialization 202 (and board reset release), the first processing unit commands the programmable logic device to generate and assert the reset lines 118, 128 of the slave processing units in step 204. The reset lines to all processors are asserted 204 after power up 202 automatically from the CPLD. Then, after a preset time, the first processor's reset line is automatically brought out of reset 206, from which it continues its bootstrapping sequence 208. The first processor can re-reset and restart 210 the bootstrapping sequence to the second, third, etc . . .processors, if for any reasons a processor fault occurs during execution or a newer bootstrap code is needed to replace the current code in the volatile memory. This can occur not only at power up. The first processing unit then commands the programmable logic device to generate and assert the reset lines 118, 128 of the slave processing units in step 212. The first processing unit 102 then loads the boot code for the slave

processing units 112, 122 from the flash device to the volatile memory devices 114, 124 using the local bus 110 in step 214. The first processing unit 102 then commands the programmable logic device 108 to de-assert the reset lines 118, 128 of the second processing units 112, 122 in step 216. The second processing units 112, 122 then accept the boot code from the volatile memory device 114, 124, respectively, and perform their bootstrap procedure in step 218. This is followed by loading of application and starting of application code in step 220. This code can be loaded through shared volatile memory or through other communication mechanisms e.g. PCI as described.

[0022] This method and system can be used for multi-drop (parallel processing) systems that have multiple slave processing units performing similar functionality and executing the same boot code. The master processing unit's firmware provides identity to each slave processing unit by posting information through shared memory. One advantage of this system is that the bootstrap procedure can be changed by simply downloading the new procedure to the first processing unit 102 over the communication port 104, wherein the new procedure is stored in the flash device 106.

[0023] Slave processing units may, however, be designed only to boot asynchronously from flash devices. Accordingly, the slave processing unit will not supply a synchronous clock output during its initial addressing of the memory devices. Another synchronous clock issue is the inability to change the slave processing unit's internal phase lock loop (PLL) to a different operation frequency

while using the volatile memory boot architecture. It is sometimes common for a processor to start at a low clock frequency and to change the PLL to a much higher operating frequency during the bootstrap sequence. When this occurs, the synchronous clock stops momentarily and then restarts again but is erratic until a determined settling time occurs. . By using the correct sequence of No-Operation instructions, as mentioned previously, and instruction cache, these processing units can bootstrap from the synchronous SRAM devices where the SRAM clock is discontinuous through the bootstrap sequence.

[0024] FIG. 3 illustrates a computer system 300 according to another embodiment of the present invention in which each second processing unit can be selectively bootstrapped from either a flash device or a volatile memory device. FIG. 3 is similar to FIG. 1 except that each processing unit 112, 122 is connected to a separate flash device 115, 125 and the volatile memory devices 114, 124, respectively. In addition, the programmable logic device 108 outputs a flash output enable signal to each flash device 115, 125.

[0025] The operation of the computer system 300 illustrated in FIG. 3 will now be described with reference to FIG. 4. First, the first processing unit 102 is bootstrapped in a known manner using the flash device 106 in step 402. The first processing unit 102 then determines which bootstrap procedure has been selected for the slave processing units in step 404. The first processing unit 102 then sets a command register in the programmable logic device 108 so that the logic units are set for the appropriate procedure in step 406. Alternatively, the selected

bootstrapping mode can be input into the programmable logic unit either manually or through processor control via a Mode Select line. If it is determined in step 408 that the flash procedure has been selected, the slave processing units are bootstrapped in a known manner using the boot code stored in the flash devices 115, 125 in step 410.

[0026] Alternatively, if it is determined in step 408 that the volatile memory procedure has been selected, the master processing unit commands the programmable logic device generate and assert the reset lines 118, 128 of the slave processing units in step 412. The first processing unit 102 then loads the boot code for the slave processing units 112, 122 from the flash device to the volatile memory devices 114, 124 using the local bus 110 in step 414. The first processing unit 102 then commands the programmable logic device 108 to de-assert the reset lines 118, 128 of the slave processing units 112, 122 in step 416. The slave processing units 112, 122 then accept the boot code from the volatile memory device 114, 124, respectively, and perform their bootstrap procedure in step 418. This is followed by loading of application and starting of application code. This code can be loaded through shared volatile memory or through other communication mechanisms e.g. PCI as described in step 420. The application load and startup details are out of the scope of this invention.

[0027] The many features and advantages of the invention are apparent from the detailed specification, and thus, it is intended by the appended claims to cover all such features and advantages of the invention which fall within the true spirits

and scope of the invention. Further, since numerous modifications and variations will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation illustrated and described, and accordingly, all suitable modifications and equivalents may be resorted to, falling within the scope of the invention.